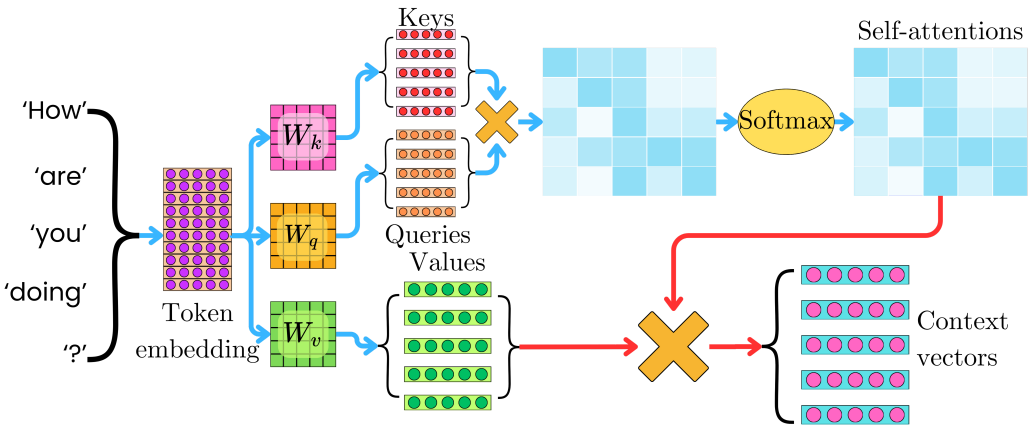
Understanding The Self-Attention

Damien Benveniste
The AiEdge



1 The Architecture

In the case of the Bahdanau/Luong attention, the goal was to capture the interactions between the tokens of the input sequence and the ones of the output sequence. In the transformer, the self-attention captures the token interactions within the sequences. It is composed of three linear layers W_k, W_q, and W_v. The input vectors to the attention layer are the internal hidden states $\mathbf{h}_i$ resulting from the model inputs. There are as many hidden states as there are tokens in the input sequence, and $\mathbf{h}_i$ corresponds to the ith token. W_k, W_q and W_v project the incoming hidden states into the so-called keys $\mathbf{k}_i$, queries $\mathbf{q}_i$ and values $\mathbf{v}_i$:

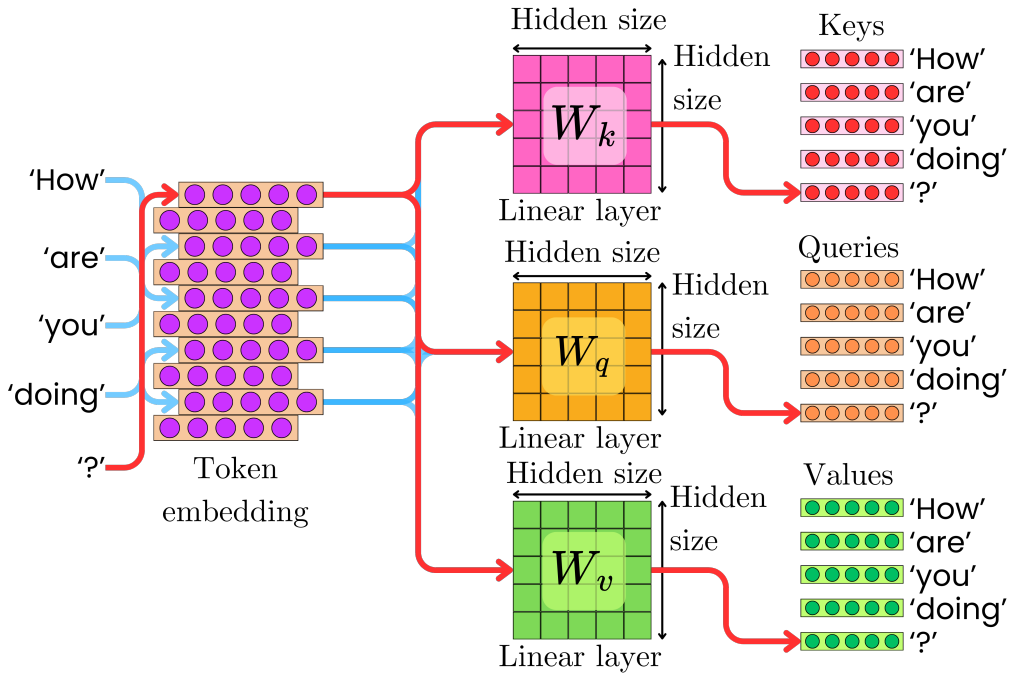
$$\mathbf{k}_i = W_k\mathbf{h}_i, \quad \text{keys} \tag{1}$$
$$\mathbf{q}_i = W_q\mathbf{h}_i, \quad \text{queries} \tag{2}$$
$$\mathbf{v}_i = W_v\mathbf{h}_i, \quad \text{values} \tag{3}$$

The keys and queries are used to compute the alignment scores:

$$e_{ij} = \frac{\mathbf{k}_i^\top \mathbf{q}_j}{\sqrt{d_{\text{model}}}} \tag{4}$$

As in the case of the Bahdanau attention, e_{ij} is the alignment score between the i^{th} word and the j^{th} word in the input sequence. d_{model} is the common naming convention for the hidden size:

$$|\mathbf{h}_i| = |\mathbf{k}_i| = |\mathbf{q}_i| = |\mathbf{v}_i| = d_{\text{model}} = \text{Hidden size} \tag{5}$$



The scaling factor $\sqrt{d_{\text{model}}}$ in the scaled dot-product is used to counteract the effect of the dot product's magnitude growing with the dimensionality d_{model}, which stabilizes gradients and ensures numerical stability during training. It is common to represent to represent those operations as matrix multiplications. With the matrix $K = [\mathbf{k}_1, \ldots, \mathbf{k}_N]$ and $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_N]$, we have:

$$E = \frac{Q^\top K}{\sqrt{d_{\text{model}}}} \tag{6}$$

or:

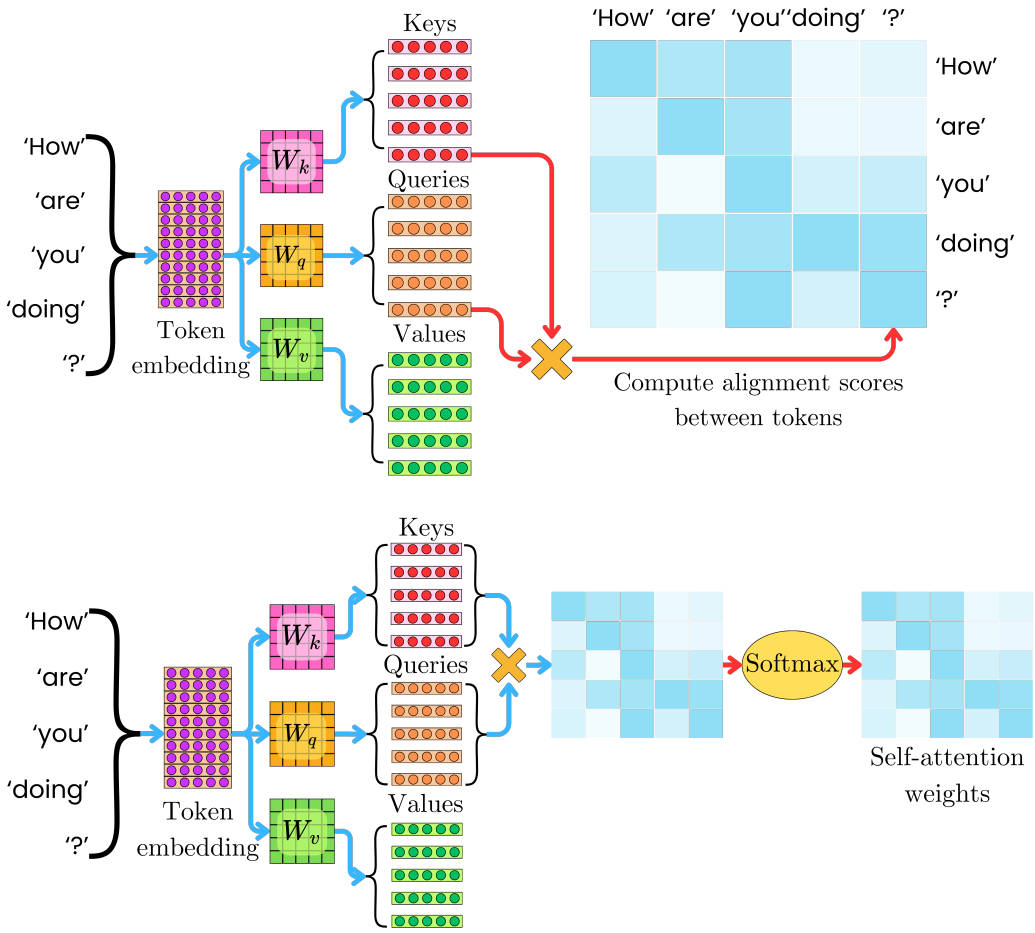
$$E = \frac{1}{\sqrt{d_{\text{model}}}} \begin{bmatrix} \mathbf{q}_1^\top \mathbf{k}_1 & \mathbf{q}_1^\top \mathbf{k}_2 & \cdots & \mathbf{q}_1^\top \mathbf{k}_N \\ \mathbf{q}_2^\top \mathbf{k}_1 & \mathbf{q}_2^\top \mathbf{k}_2 & \cdots & \mathbf{q}_2^\top \mathbf{k}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_N^\top \mathbf{k}_1 & \mathbf{q}_N^\top \mathbf{k}_2 & \cdots & \mathbf{q}_N^\top \mathbf{k}_N \end{bmatrix} \tag{7}$$

with N being the number of tokens in the sequence.

As for the other attentions, the alignment scores are normalized to 1 through a Softmax transformation:

$$a_{ij} = \text{Softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{j=1}^{N} \exp(e_{ij})} \tag{8}$$

where a_{ij} is the attention weight between the tokens i and j, quantifying how strongly the model should attend to token j when processing token i. Because we have $\sum_{j=1}^{N} a_{ij} = 1$, a_{ij} can be interpreted as the probability that token j is relevant to token i.



The attention weights are used to compute a weighted average of the values vectors:

$$\mathbf{c}_i = \sum_{j=1}^{N} a_{ij}\mathbf{v}_j \tag{9}$$

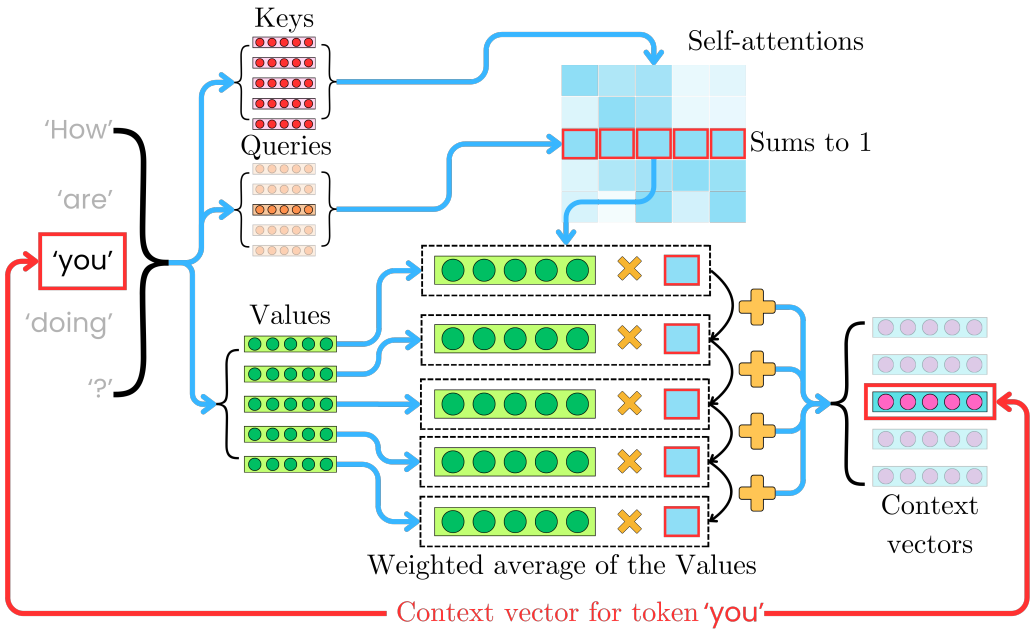
$\mathbf{c}_i$ are the context vectors coming out of the attention layer, but they are also referred to as another intermediary set of hidden states within the network. Using the more common matrix notation, we have:

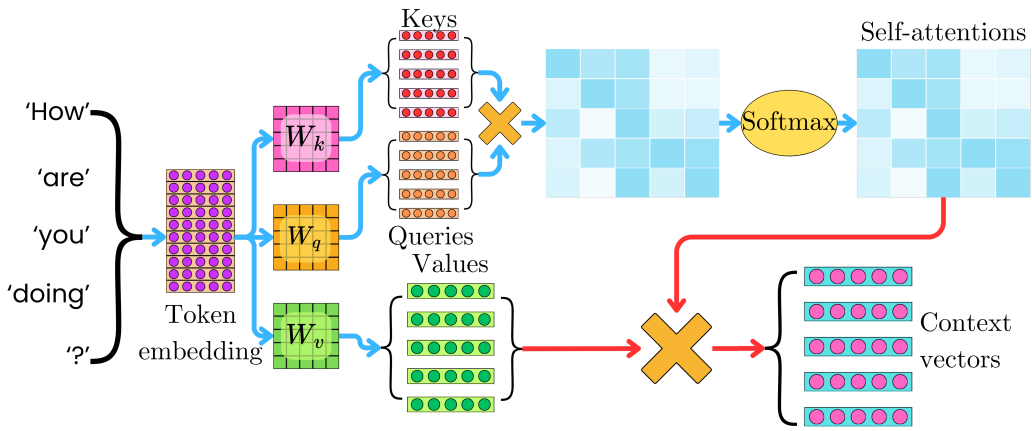
$$C = AV^\top \tag{10}$$

where $V = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$, $C = [\mathbf{c}_1, \ldots, \mathbf{c}_N]$ and $A = \text{Softmax}(E)$ is the matrix of attention weights.

The whole set of computations happening in the attention layer can be summarized as the following equation:

$$C = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_{\text{model}}}}\right)V^\top \tag{11}$$





2 The Keys, Queries, and Values Naming Convention

The names "queries", "keys", and "values" are inspired by information retrieval systems (like databases or search engines). Each token generates a query, key, and value to "retrieve" relevant context from other tokens. The model learns to dynamically search for relationships between tokens.

The queries represent what the current token is "asking for." For example, the word *"it"* in *"The cat sat because it was tired,"* the query seeks antecedents (e.g., *"cat"*). The keys represent what other tokens "offer" as context. In our example, the key for *"cat"* signals it is a candidate antecedent for *"it."* The values are the actual content to aggregate based on attention weights. The value for *"cat"* encodes its contextual meaning (e.g., entity type, role

in the sentence, ...). For each query (current token), the model "retrieves" values (context) by comparing the query to all keys (other tokens). For example, let us consider the sentence:

"The bank is steep, so it's dangerous to stand near it."

- Query (*"it"*): "What does *'it'* refer to?"
- Keys (*"bank," "steep," "dangerous"*): Highlight candidates for reference.
- Values: Encode the meaning of each candidate.

The model computes high attention weights between the query (*"it"*) and keys (*"bank," "steep"*), then aggregates their values to infer *"it"* refers to the riverbank.